Лабораторная работа № 1. Julia. Установка и настройка. Основные принципы.

Абакумова Олеся Максимовна, НФИбд-02-22

Содержание

Список иллюстраций

Список таблиц

# 1 Цель работы

Основная цель работы – подготовить рабочее пространство и инструментарий для работы с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.

# 2 Задание

1. Установите под свою операционную систему Julia, Jupyter.
2. Используя Jupyter Lab, повторите примеры из разделов.
3. Выполните задания для самостоятельной работы.

# 3 Выполнение лабораторной работы

## 3.1 Подготовка инструментария к работе

Для выполнения данной лабораторной работы нам необходимы Julia и Jupyter. Так, как они установлены у меня уже изначально, то могу продемонстрировать их наличие на моей ОС (рис. 1):

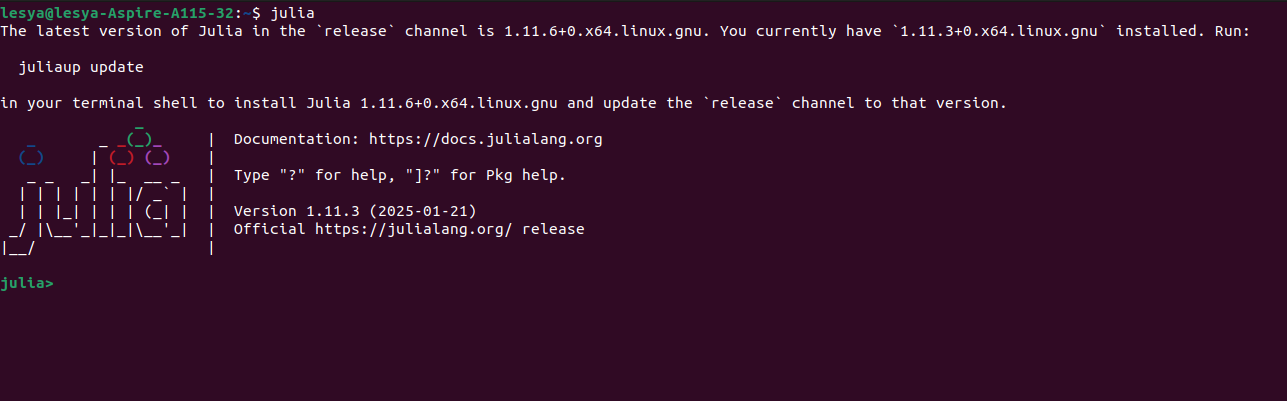


Рис. 1: Запуск Julia

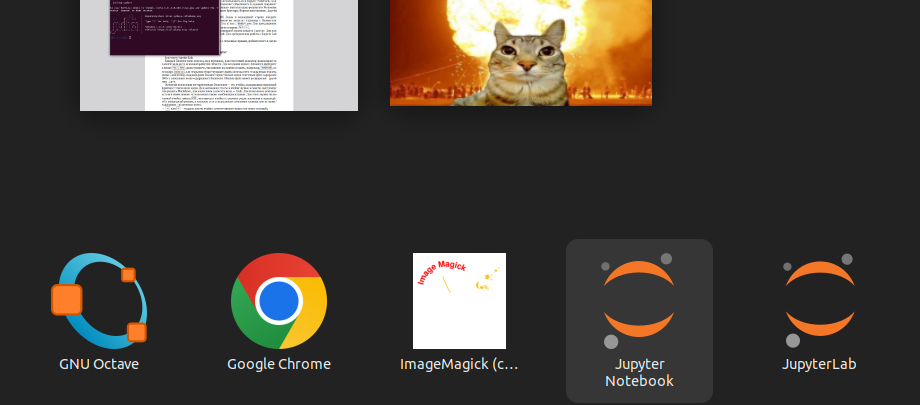


Рис. 2: Запуск Jupyter

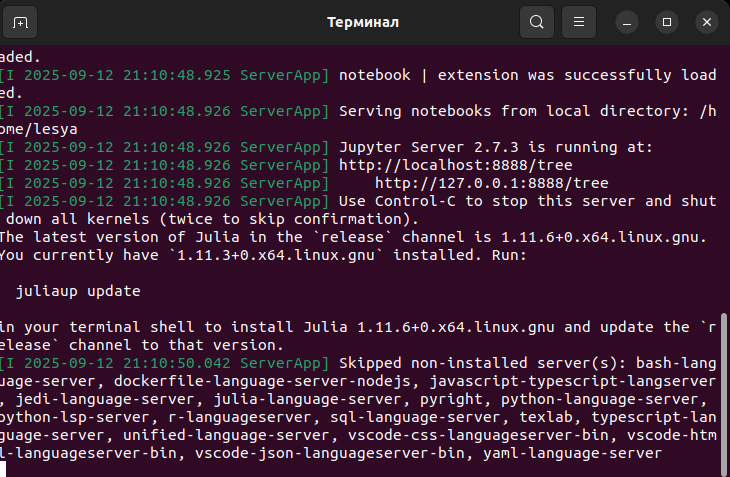


Рис. 3: Запуск Jupyter

Зайдя в Jupyter Notebook, создадим блокнот для работы с Julia (рис. 4):

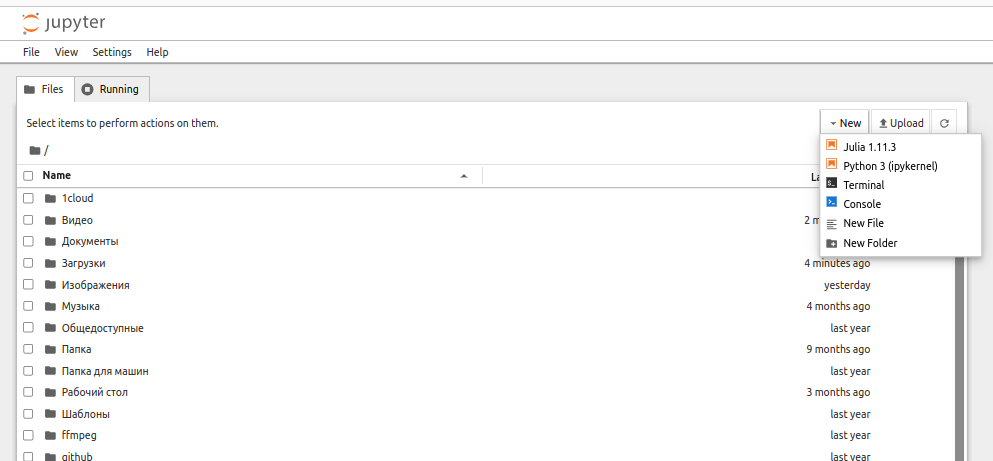


Рис. 4: Создание нового блокнота для Julia

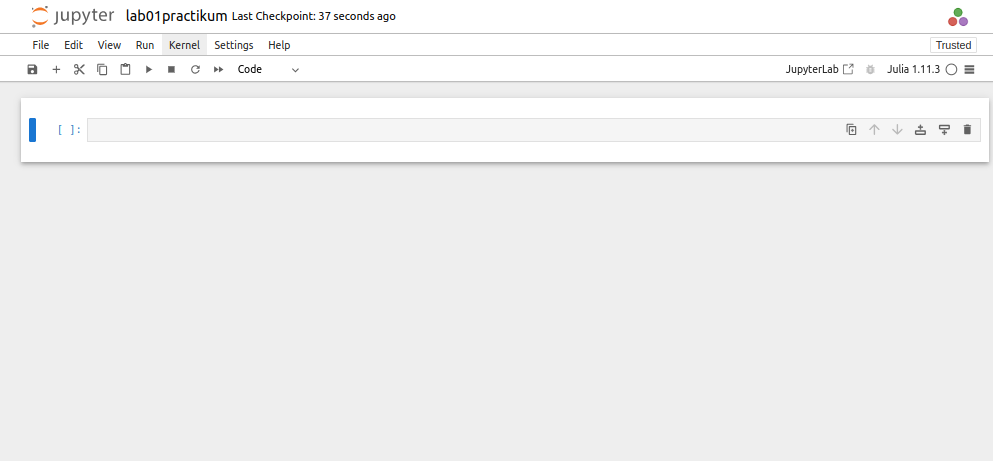


Рис. 5: Блокнот имеет ядро Julia

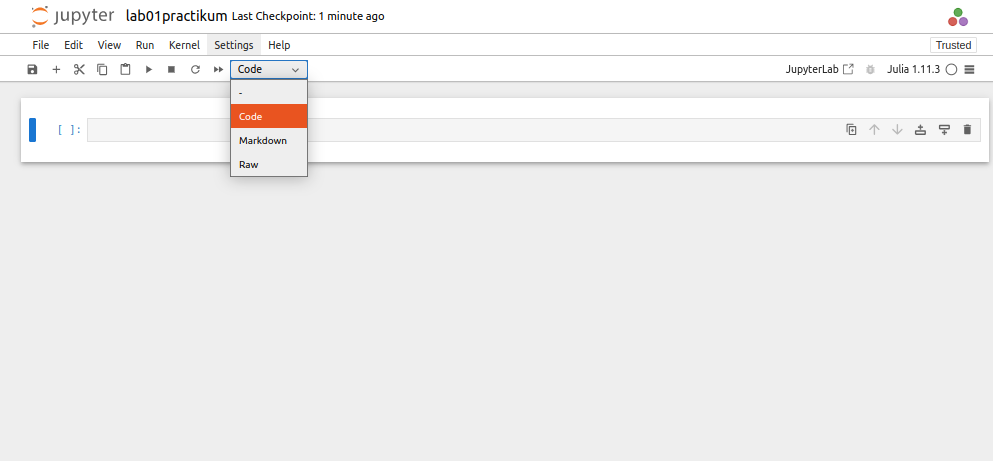


Рис. 6: Режимы вставки ячейки

## 3.2 Основы работы в блокноте Jupyter

Каждый блокнот (или консоль, или терминал, или текстовый редактор) располагается в своей вкладке в основной рабочей области. Для создания нового блокнота выбирается в меню File (рис. 4) , далее указывается, что именно мы хотим создать. Для открытия существующего файла используютс стандартные пункты меню и навигатор. Каждый файл-блокнот представляет собой текстовый файл в формате JSON с описанием всего содержимого блокнота. Обычно файл имеет расширение .ipynb или .ipyn. Основная концепция интерактивных блокнотов — это ячейка, содержащая отдельный фрагмент текста (или кода). Для написания текста в ячейке нужно в панели инструментов указать Markdown, для написания элемента кода – Code (рис. 6). Для изменения режимов вставки ячеек можно использовать также комбинации клавиш. Для этого нужно на активной ячейке нажать **ESC** , что выведет ячейку из режима редактирования и переведёт её в командный режим, в котором есть специальные сочетания клавиш для вставки/вырезания/изменения ячеек:

* **a** или **b** – создать новую ячейку соответственно выше или ниже текущей;
* **x** – удалить ячейку;
* **z** – отмена удаления ячейки;
* **m** – перевести ячейку в режим текста;
* **y** – перевести ячейку в режим набора кода.

Для выполнения кода внутри ячейки выберите эту ячейку и нажмимается **Shift + Enter** или кнопку со значком **Run** на панели инструментов. Если ячейка содержит несколько строк кода, то при выполнении этой ячейки отобразится только результат последней строки (операции). Вывод результата можно подавить, завершив строку знаком «точка с запятой».

Выполним примеры кода простейшей операцией сложения в блокноте Jupyter (рис. 7):

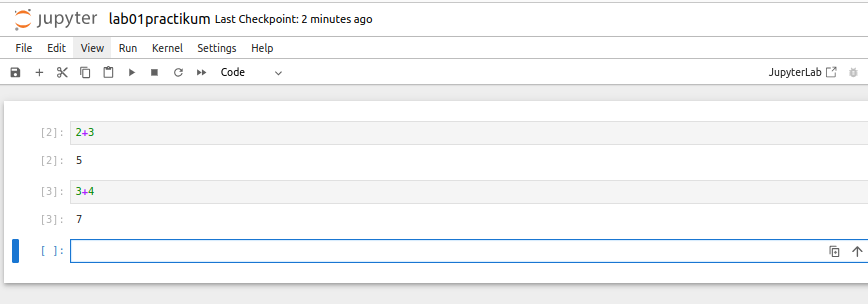


Рис. 7: Простейшие операции на языке Julia в Jupyter Lab

Если необходимо получить информацию по работе с какой-то незнакомой для вас функцией Julia, то можно поставить в ячейке перед названием этой функции знак вопроса (рис. 8):

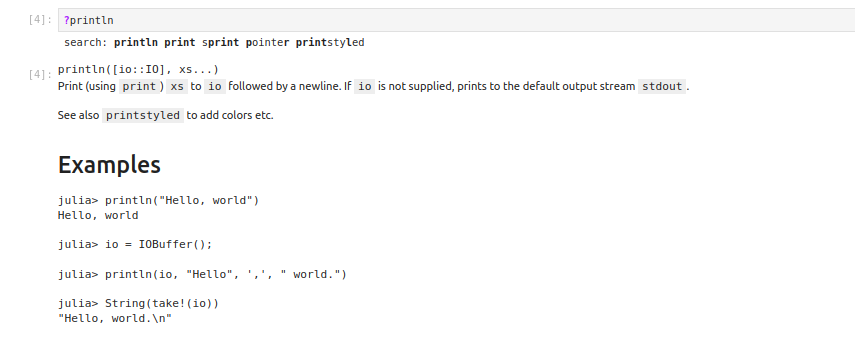


Рис. 8: Пример получения информации по функции println на языке Julia в Jupyter Lab

Если требуется использовать команды из командной оболочки операционной системы, то перед соответствующей командой нужно поставить знак «точка с запятой». Например, для пользователей ОС Linux можно вывести текущую дату и имя пользователя, используя последовательно команды date и whoami. Для пользователей других ОС следует использовать команды оболочки соответствующей операционной системы (рис. 9):

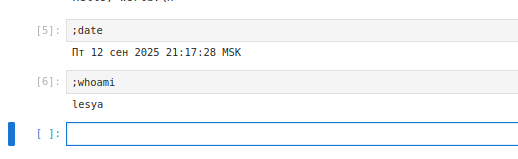


Рис. 9: Пример получения информации о дате и пользователе ОС Linux в Jupyter Lab

Для очистки результатов выполнения ячеек следует использовать меню **Edit**(рис. 10):

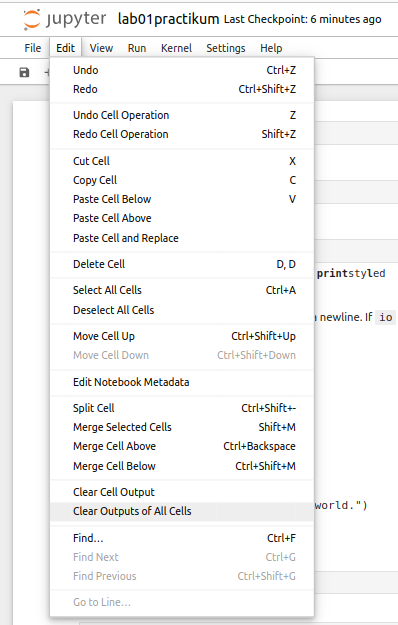


Рис. 10: Очистка результатов выполнения ячеек

## 3.3 Основы синтаксиса Julia на примерах

Далее приведены простейшие примеры с использованием синтаксиса Julia, выполненные в блокноте Jupyter Lab. Определение типа числовой величины можно реализовать с помощью команды **typeof(Number)** (рис. 11). Здесь Number – конкретное число, например, или , или числовой результат какой-либо операции, например,, , значение числа . В Julia введены специальные значения **Inf**, **-Inf**, **NaN**, обозначающие бесконечность и отсутствие какого-либо значения. Такие значения могут получаться в результате операций типа деления на ноль, а также могут быть допустимой частью выражений, поскольку в языке имеют тип вещественного числа. Для определения крайних значений диапазонов целочисленных числовых величин можно воспользоваться следующим кодом (рис. 11):

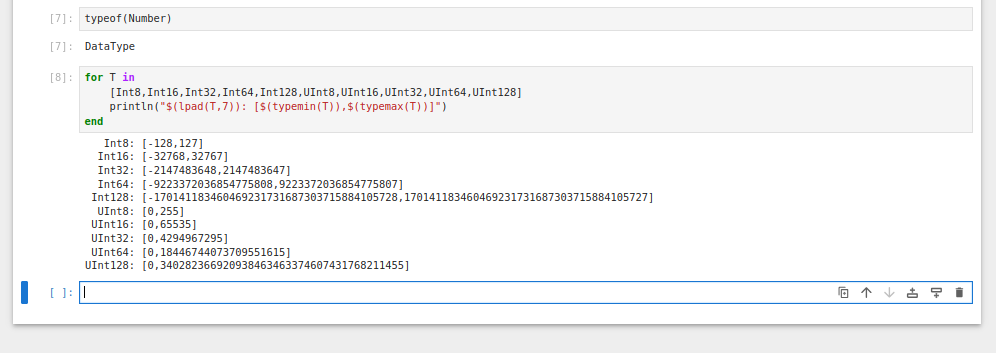


Рис. 11: Определение крайних значений диапазонов целочисленных числовых величин и определение типа числовой величины

В результате получим минимальные и максимальные значения целочисленных типов.

В Julia преобразование типов можно реализовать или прямым указанием, например вещественное число 2.0 преобразовать в целое, а число 2 в символ: **Int64(2.0)**, **Char(2)** или использовать обобщённый оператор преобразования типов **convert()**, например: **convert(Int64, 2.0)**, **convert(Char,2)** Преобразование 1 в булевое **true**, 0 – в булевое **false**: **Bool(1)**, **Bool(0)**. Для приведения нескольких аргументов к одному типу, если это возможно, используется оператор **promote()**, например: **promote(Int8(1)**, **Float16(4.5)**, **Float32(4.1))** В данном выражении все аргументы оператора **promote()** в результате будут иметь тип **Float32**, в чём можно убедиться, воспользовавшись функцией определения типа **typeof** (рис. 12):

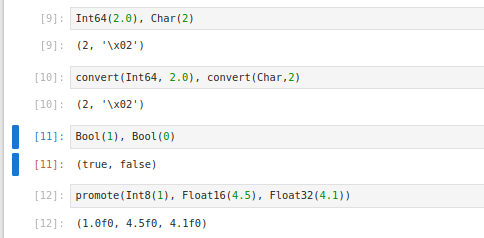


Рис. 12: Примеры приведения аргументов к одному типу

Базовый синтаксис определения функции:

function <Имя> (<СписокПараметров>)  
 <Действия>  
end

Например, определим функцию возведения переменной в квадрат и возведём в квадрат число 4 (рис. 13):

function f(x)  
 x^2  
end  
f(4)

Другой способ определения несложных функций (рис. 14:

<Имя> (<СписокПараметров>) = <Выражение>

Пример определения одномерных массивов (вектор-строка и вектор-столбец) и обращение к их вторым элементам (рис. 13):

a = [4 7 6] # вектор-строка  
b = [1, 2, 3] # вектор-столбец  
a[2], b[2] # вторые элементы векторов a и b

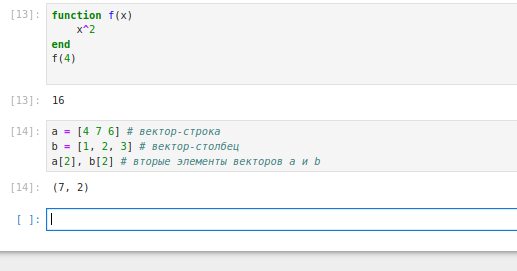


Рис. 13: Пример определения одномерных массивов и пример определения функций

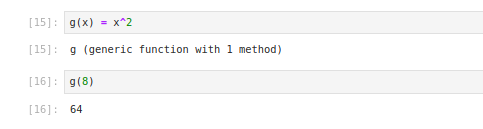


Рис. 14: Пример определения функций

Пример определения двумерного массива (матрицы) и обращение к его элементам (рис. 15):

a = 1; b = 2; c = 3; d = 4 # присвоение значений  
Am = [a b; c d] # матрица 2 х 2  
Am[1,1], Am[1,2], Am[2,1], Am[2,2] # элементы матрицы

Пример выполнения операций над массивами (aa’ – транспонирование вектора) (рис. 15):

aa = [1 2]  
AA = [1 2; 3 4]  
aa\*AA\*aa

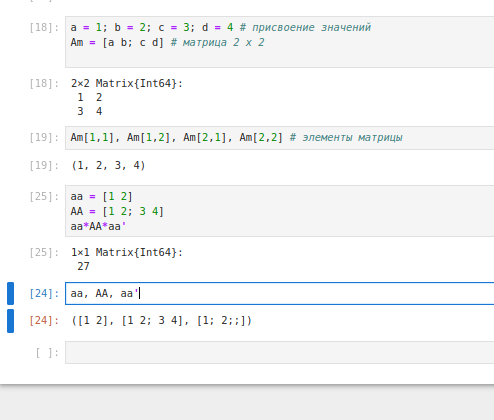


Рис. 15: Примеры работы с массивами

## 3.4 Задания для самостоятельной работы

1. Изучите документацию по основным функциям Julia для чтения / записи / вывода информации на экран: **read(), readline(), readlines(), readdlm(), print(), println(), show(), write()**. Приведите свои примеры их использования, поясняя особенности их применения.

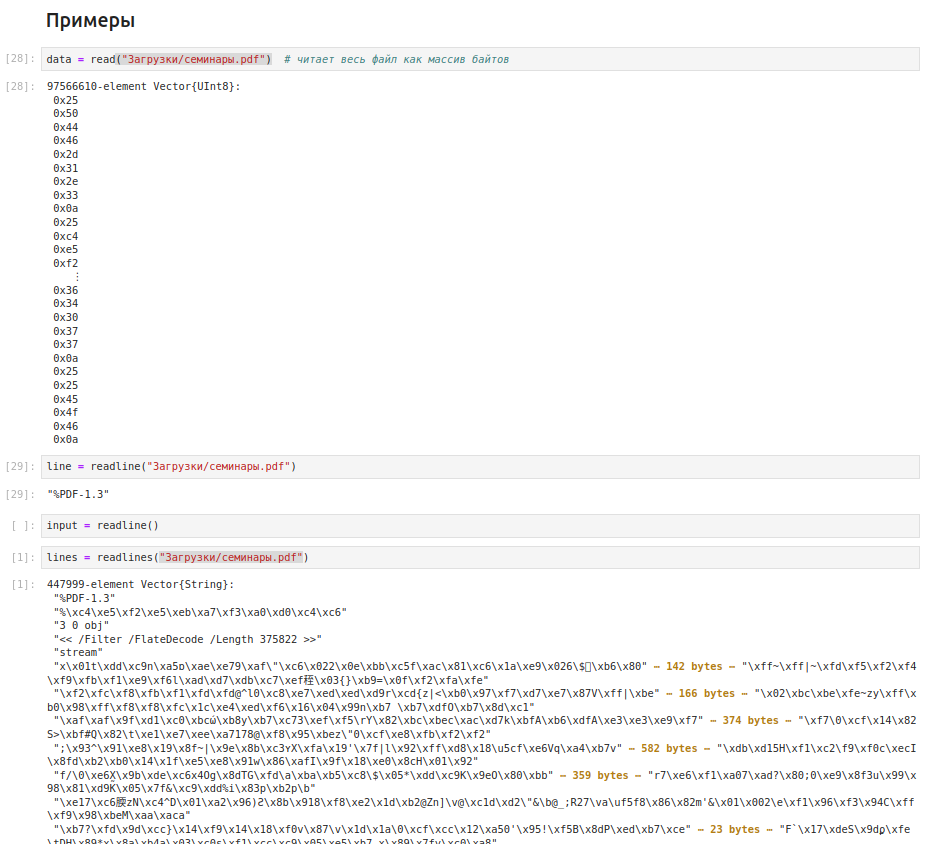


Рис. 16: Функции read(), readline(), readlines()

* **read()** читает весь файл, как массив байтов;
* **readline()** читает только одну строку;
* **readlines()** читает все строки.

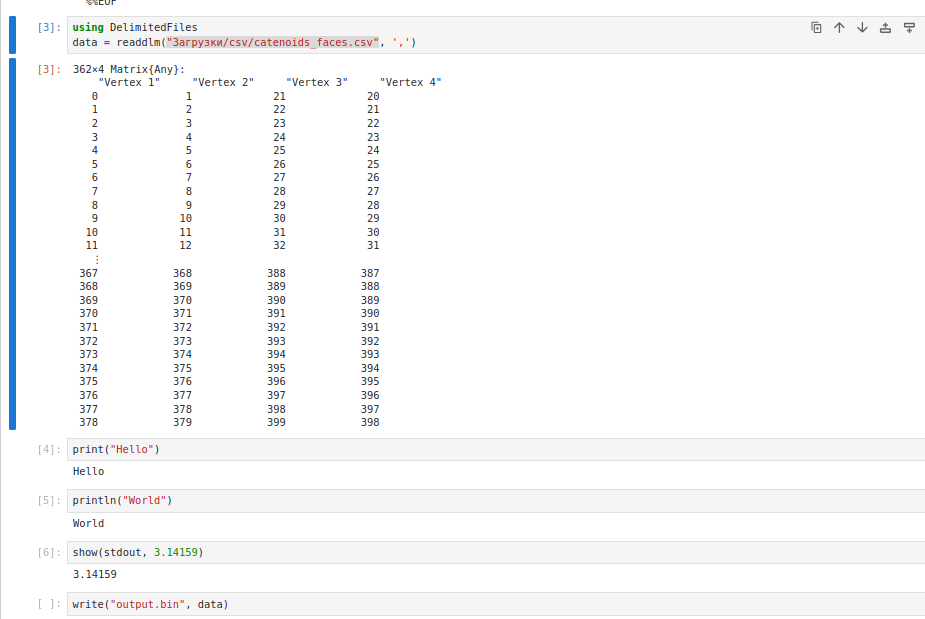


Рис. 17: Функции readlm(), print(), println(), show(), write()

* **readlm()** читает с разделителем;
* **print()** вывод текста по частям;
* **println()** вывод в отдельных строках.
* **show()** выводит текстовые данные;
* **write()** выводит двоичные данные.

1. Изучите документацию по функции **parse()**. Приведите свои примеры её использования, поясняя особенности её применения.

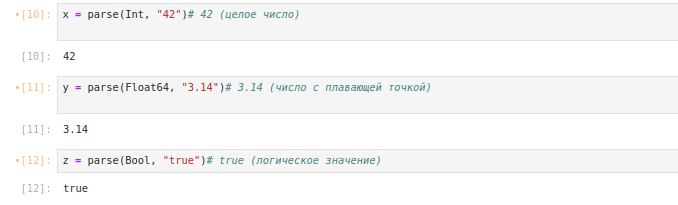


Рис. 18: Функция parse()

* **parse()** используется для преобразования строки в число определенного типа

1. Изучите синтаксис Julia для базовых математических операций с разным типом переменных: сложение, вычитание, умножение, деление, возведение в степень, извлечение корня, сравнение, логические операции. Приведите свои примеры с пояснениями по особенностям их применения.

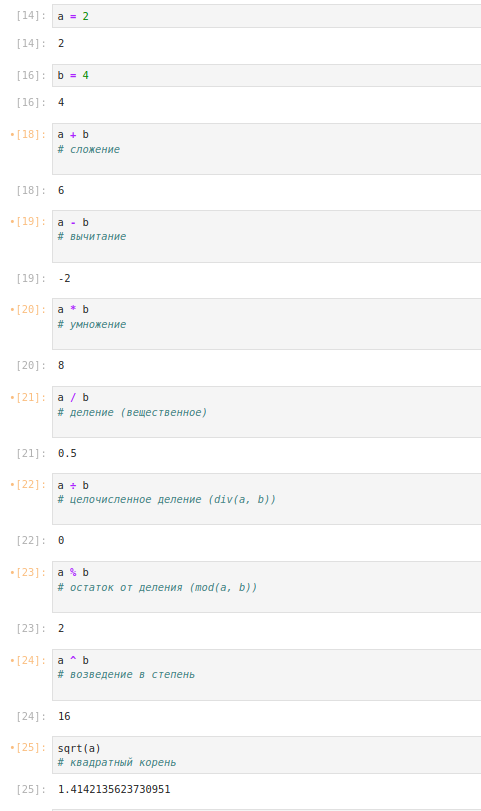


Рис. 19: Математические операции с разным типом переменных

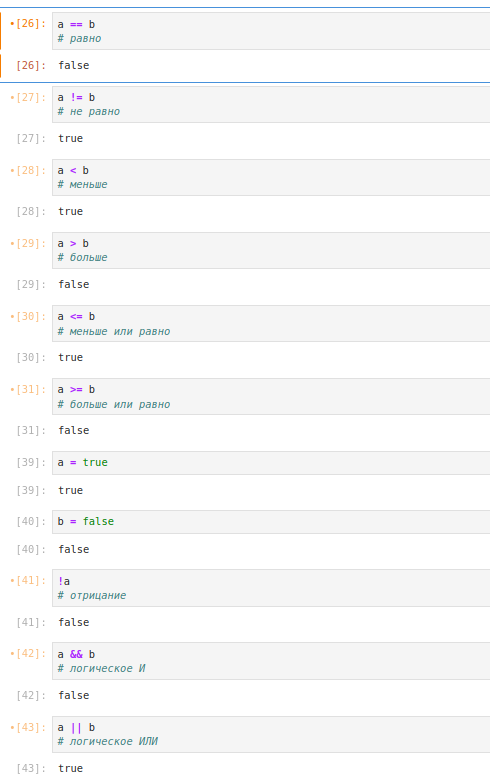


Рис. 20: Математические операции с разным типом переменных(логические операции)

1. Приведите несколько своих примеров с пояснениями с операциями над матрицами и векторами: сложение, вычитание, скалярное произведение, транспонирование, умножение на скаляр.



Рис. 21: Операции над матрицами и векторами

# 4 Выводы

В процессе выполнения данной лабораторной работы подготовила рабочее пространство и инструментарий для с языком программирования Julia, на простейших примерах познакомиться с основами синтаксиса Julia.